frivol

Generated by Doxygen 1.8.1.2

Thu May 30 2013 02:39:00

Contents

Chapter 1

Class Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Class Documentation

3.1 frivol::fortune::Algorithm< PolicyT > Class Template Reference

`#include <fortune.hpp>`

Public Types

- typedef PolicyT::Coord **CoordT**
- typedef Point< CoordT > **PointT**
- typedef VoronoiDiagram< CoordT > **VoronoiDiagramT**

Public Member Functions

- Algorithm (const containers::Array< PointT > &sites)
- void step ()

  Runs the algorithm one event handling forward.
- CoordT getSweeplineY () const
- bool isFinished ()

  Returns true if the algorithm has finished.
- void finish ()

  Steps the algorithm until the end.
- int getVoronoiVertexCount () const

  Returns the number of Voronoi vertices met in the algorithm.
- const VoronoiDiagramT & getVoronoiDiagram () const

3.1.1 Detailed Description

template<typename PolicyT = DefaultPolicy>class frivol::fortune::Algorithm< PolicyT >

State of Fortune's algorithm.

Template Parameters

PolicyT	The algorithm policy to use, instance of Policy template.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 template<typename PolicyT > frivol::fortune::Algorithm< PolicyT >::Algorithm (const containers::Array< PointT > & *sites*)

Constructs algorithm state.

Parameters

points	Reference to the input set of sites. The object must exist throughout the existence of the Algorithm.

3.1.3 Member Function Documentation

3.1.3.1 template<typename PolicyT > PolicyT::Coord frivol::fortune::Algorithm< PolicyT >::getSweeplineY ( ) const

Returns the sweepline Y coordinate of last step(). Undefined return value if step() has not been called yet.

3.1.3.2 template<typename PolicyT > const VoronoiDiagram< typename PolicyT::Coord > & frivol::fortune::Algorithm< PolicyT >::getVoronoiDiagram ( ) const

Returns the Voronoi diagram constructed in the algorithm. The diagram is complete if the algorithm is finished.

The documentation for this class was generated from the following files:

- /home/topi/unison/Asiakirjat/frivol/frivol/fortune/fortune.hpp
- /home/topi/unison/Asiakirjat/frivol/frivol/fortune/fortune_impl.hpp

3.2 frivol::fortune::Arc Struct Reference

Information of an arc in BeachLine.

`#include <beach_line.hpp>`

Public Attributes

- Idx site

  The index of the site from which the arc originates.
- Idx arc_id

  The ID of the arc.

The documentation for this struct was generated from the following file:

- /home/topi/unison/Asiakirjat/frivol/frivol/fortune/beach_line.hpp

3.3 frivol::containers::Array< T > Class Template Reference

`#include <array.hpp>`

Public Member Functions

- Array ()

  Creates an array with size 0.

- Array (Idx size)
- Idx getSize () const

  Returns the size of the array.
- void resize (Idx size)

- const T & operator[] (Idx index) const
- T & operator[] (Idx index)

3.3.1 Detailed Description

template<typename T>class frivol::containers::Array< T >

Simple fixed-size array.

Template Parameters

T	The type of stored elements. Should be default constructible.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 template<typename T > frivol::containers::Array< T >::Array (Idx *size*)

Creates an array with all elements default-constructed.

Parameters

size	The size of the array.

3.3.3 Member Function Documentation

3.3.3.1 template<typename T > const T & frivol::containers::Array< T >::operator[] (Idx *index*) const

Returns reference to an element in the array.

Parameters

index	The zero-based index of the element.

Exceptions

std::out_of_range	if FRIVOL_ARRAY_BOUNDS_CHECKING is defined and 'index' overflows.

3.3.3.2 template<typename T > T & frivol::containers::Array< T >::operator[] (Idx *index*)

Returns reference to an element in the array.

Parameters

index	The zero-based index of the element.

Exceptions

std::out_of_range	if FRIVOL_ARRAY_BOUNDS_CHECKING is defined and 'index' overflows.

3.3.3.3 template<typename T > void frivol::containers::Array< T >::resize (Idx *size*)

Resizes the array to size. If size decreases the extra elements are removed. If size increases, the new elements are default-constructed. The operation may assign the current elements to a new place, and therefore pointers to the array may be invalidated.

Parameters

size	The new size.

The documentation for this class was generated from the following files:

- /home/topi/unison/Asiakirjat/frivol/frivol/containers/array.hpp
- /home/topi/unison/Asiakirjat/frivol/frivol/containers/array_impl.hpp

3.4 frivol::fortune::BeachLine< PolicyT > Class Template Reference

`#include <beach_line.hpp>`

Public Types

- typedef PolicyT::Coord **CoordT**
- typedef Point< CoordT > **PointT**

Public Member Functions

- BeachLine (const containers::Array< PointT > &sites, Idx max_arcs)
- Idx getMaxArcCount () const
- Idx insertArc (Idx site, const CoordT &sweepline_y)
- void removeArc (Idx arc_id)
- Idx getLeftArc (Idx arc_id)
- Idx getRightArc (Idx arc_id)
- Idx getLeftmostArc ()

  Returns the ID of the leftmost arc, or nil_idx if the beach line is empty.
- Idx getRightmostArc ()

  Returns the ID of the rightmost arc, or nil_idx if the beach line is empty.
- Idx getOriginSite (Idx arc_id)

3.4.1 Detailed Description

template<typename PolicyT>class frivol::fortune::BeachLine< PolicyT >

The advancing sweepline of Fortune's algorithm. Consists of parabolic arcs that are defined to be the curves that have equal distances from an input site and from the sweepline. Adjacent arcs are separated by their intersection points called breakpoints.

The arcs of the beach line are identified by numerical arc IDs. The maximum number of arcs in the beach line must be specified in advance.

Template Parameters

PolicyT	The algorithm policy to use, instance of Policy template.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 template<typename PolicyT > frivol::fortune::BeachLine< PolicyT >::BeachLine (const containers::Array< PointT > & *sites*, Idx *max_arcs*)

Constructs BeachLine.

Parameters

sites	The input sites for the algorithm.
max_arcs	The number of arcs the beach line must be able to contain.

3.4.3 Member Function Documentation

3.4.3.1 template<typename PolicyT > Idx frivol::fortune::BeachLine< PolicyT >::getLeftArc (Idx *arc_id*)

Returns the ID of the arc left from given arc.

Parameters

arc_id	ID of the arc.

Returns

arc ID of the arc to the left from arc_id, or nil_idx if arc_id is the leftmost arc.

3.4.3.2 template<typename PolicyT > Idx frivol::fortune::BeachLine< PolicyT >::getMaxArcCount ( ) const

Gets the maximum number of arcs there can be in the beach line. The arc IDs are in 0, ..., getMaxArcCount()-1.

3.4.3.3 template<typename PolicyT > Idx frivol::fortune::BeachLine< PolicyT >::getOriginSite (Idx *arc_id*)

Returns the index of the origin site of given arc.

Parameters

arc_id	ID of the arc.

3.4.3.4 template<typename PolicyT > Idx frivol::fortune::BeachLine< PolicyT >::getRightArc (Idx *arc_id*)

Returns the ID of the arc right from given arc.

Parameters

arc_id	ID of the arc.

Returns

arc ID of the arc to the right from arc_id, or nil_idx if arc_id is the rightmost arc.

3.4.3.5 template<typename PolicyT > Idx frivol::fortune::BeachLine< PolicyT >::insertArc (Idx *site*, const CoordT & *sweepline_y*)

Adds arc to the right place in the beach line. If the beach line is nonempty, the arc under the new arc is split split in two, so that the original arc is on the right from the new arc and an additional arc is created to the left from the new arc.

Parameters

site	The origin site of the arc. The arc is placed in the position of the X-coordinate of the site.
sweepline_y	The Y-coordinate of the sweep line that defines the parabolas.

Returns

the ID of the new arc.

Exceptions

std::logic_error	if the maximum number of arcs (getMaxArcCount()) are already in use.

3.4.3.6 template<typename PolicyT > void frivol::fortune::BeachLine< PolicyT >::removeArc (Idx *arc_id*)

Removes arc from the beach line.

Parameters

arc_id	The ID of the arc to remove.

The documentation for this class was generated from the following files:

- /home/topi/unison/Asiakirjat/frivol/frivol/fortune/beach_line.hpp
- /home/topi/unison/Asiakirjat/frivol/frivol/fortune/beach_line_impl.hpp

3.5 frivol::containers::priority_queues::DummyPriorityQueue< PriorityT > Class Template Reference

Simple implementation of PriorityQueueConcept.

`#include <dummy_priority_queue.hpp>`

Public Member Functions

- **DummyPriorityQueue** (Idx size)
- std::pair< Idx, PriorityT > **pop** ()
- bool **empty** () const
- void **setPriority** (Idx key, PriorityT priority)
- void **setPriorityNIL** (Idx key)

The documentation for this class was generated from the following files:

- /home/topi/unison/Asiakirjat/frivol/frivol/containers/priority_queues/dummy_priority_queue.hpp
- /home/topi/unison/Asiakirjat/frivol/frivol/containers/priority_queues/dummy_priority_queue_impl.hpp

3.6 frivol::containers::search_trees::DummySearchTree< ElementT > Class Template Reference

Simple implementation of SearchTreeConcept (a wrapper around std::list).

`#include <dummy_search_tree.hpp>`

Public Types

- typedef std::list< ElementT >
  ::iterator **Iterator**

Public Member Functions

- template<typename FuncT >
  Iterator **search** (FuncT func)

The documentation for this class was generated from the following file:

- /home/topi/unison/Asiakirjat/frivol/frivol/containers/search_trees/dummy_search_tree.hpp

3.7 frivol::containers::DynamicArray< T > Class Template Reference

`#include <dynamic_array.hpp>`

Public Member Functions

- DynamicArray ()
  Creates a dynamic array of size 0.
- DynamicArray (Idx size)
- Idx getSize () const
  Returns the size of the array.
- Idx add (const T &element)

- const T & operator[] (Idx index) const
- T & operator[] (Idx index)

3.7.1 Detailed Description

template<typename T>class frivol::containers::DynamicArray< T >

Array that is more efficient at adding elements to the end than a regular array because DynamicArray constructs more elements in advance.

Template Parameters

T	The type of stored elements. Should be default constructible.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 template<typename T > frivol::containers::DynamicArray< T >::DynamicArray (Idx *size*)

Creates a dynamic array with all elements default-constructed.

Parameters

size	The size of the array.

3.7.3 Member Function Documentation

3.7.3.1 template<typename T> Idx frivol::containers::DynamicArray< T >::add (const T & *element*)

Adds given element to the end of the dynamic array, increasing its size by one.

Parameters

element	The element to add.

Returns

the index of the added element.

3.7.3.2 template<typename T > const T & frivol::containers::DynamicArray< T >::operator[] (Idx *index*) const

Returns reference to an element in the array.

Parameters

index	The zero-based index of the element.

Exceptions

std::out_of_range	if FRIVOL_ARRAY_BOUNDS_CHECKING is defined and 'index' overflows.

3.7.3.3 template<typename T > T & frivol::containers::DynamicArray< T >::operator[] (Idx *index*)

Returns reference to an element in the array.

Parameters

index	The zero-based index of the element.

Exceptions

std::out_of_range	if FRIVOL_ARRAY_BOUNDS_CHECKING is defined and 'index' overflows.

The documentation for this class was generated from the following files:

- /home/topi/unison/Asiakirjat/frivol/frivol/containers/dynamic_array.hpp
- /home/topi/unison/Asiakirjat/frivol/frivol/containers/dynamic_array_impl.hpp

3.8 frivol::GeometryTraits< CoordT > Struct Template Reference

`#include <geometry_traits.hpp>`

3.8.1 Detailed Description

template<typename CoordT>struct frivol::GeometryTraits< CoordT >

Traits class that gives needed geometry operations for the algorithm. Implemented traits are required by Policy.

Template Parameters

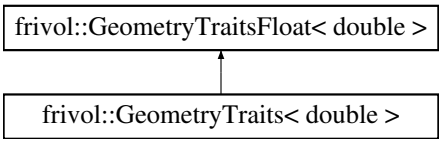
CoordT	The coordinate type to use.

The documentation for this struct was generated from the following file:

- /home/topi/unison/Asiakirjat/frivol/frivol/geometry_traits.hpp

3.9 frivol::GeometryTraits< double > Struct Template Reference

Inheritance diagram for frivol::GeometryTraits< double >:



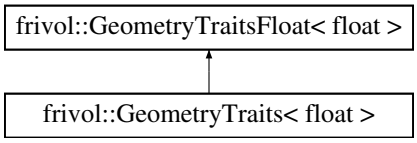
Additional Inherited Members

The documentation for this struct was generated from the following file:

- /home/topi/unison/Asiakirjat/frivol/frivol/geometry_traits.hpp

3.10 frivol::GeometryTraits< float > Struct Template Reference

Inheritance diagram for frivol::GeometryTraits< float >:



Additional Inherited Members

The documentation for this struct was generated from the following file:

- /home/topi/unison/Asiakirjat/frivol/frivol/geometry_traits.hpp

3.11 frivol::GeometryTraitsFloat< CoordT > Struct Template Reference

`#include <geometry_traits.hpp>`

Public Types

- typedef Point< CoordT > **PointT**

Static Public Member Functions

- static CoordT **getBreakpointX** (const PointT &a, const PointT &b, CoordT topy, bool positive_big)
- static PointT **getCircumcenter** (const PointT &a, const PointT &b, const PointT &c)
- static CoordT **getCircumcircleTopY** (const PointT &a, const PointT &b, const PointT &c)
- static bool **isCCW** (const PointT &a, const PointT &b, const PointT &c)

Static Public Attributes

- static constexpr CoordT **epsilon** = 1e-30

3.11.1 Detailed Description

template<typename CoordT>struct frivol::GeometryTraitsFloat< CoordT >

Implementation of GeometryTraits for floating point coordinate types (float and double).

The documentation for this struct was generated from the following files:

- /home/topi/unison/Asiakirjat/frivol/frivol/geometry_traits.hpp
- /home/topi/unison/Asiakirjat/frivol/frivol/geometry_traits_impl.hpp

3.12 frivol::GeometryTraitsImplementedConcept< CoordT > Class Template Reference

`#include <geometry_traits.hpp>`

3.12.1 Detailed Description

template<typename CoordT>class frivol::GeometryTraitsImplementedConcept< CoordT >

Concept for checking that all required GeometryTraits are implemented for given coordinate type. Required operations are:

- CoordT getBreakpointX(Point<CoordT> a, Point<CoordT> b, CoordT topy, bool positive_big) returns the X coordinate of intersection of the two parabolas defined by (x-a.x)^2 + (y-a.y)^2 = (y-topy)^2 = (x-b.x)^2 + (y-b.y)^2 The function may assume that a.x <= b.x, a.y <= topy and b.y <= topy. The function should choose the solution where the parabola around a goes under the parabola around b. In cases where this does not happen, the result should be very big number, positive if positive_big, otherwise negative.
- Point<CoordT> getCircumcenter(Point<CoordT> a, Point<CoordT> b, Point<CoordT> c) returns the center point of the circumscribed circle around triangle 'abc'.
- CoordT getCircumcircleTopY(Point<CoordT> a, Point<CoordT> b, Point<CoordT> c) returns the Y coordinate of the top point (i.e. highest Y coordinate) of the circumscribed circle around triangle 'abc'.

- bool isCCW(Point<CoordT> a, Point<CoordT> b, Point<CoordT> c) returns true if triangle 'abc' is oriented counterclockwise.

Template Parameters

CoordT	The coordinate type.

The documentation for this class was generated from the following file:

- /home/topi/unison/Asiakirjat/frivol/frivol/geometry_traits.hpp

3.13 frivol::Point< CoordT > Struct Template Reference

`#include <point.hpp>`

Public Member Functions

- Point (CoordT x, CoordT y)
- Point ()

  Constructs point with undefined values as coordinates.

Public Attributes

- CoordT x

  The X coordinate of the point.
- CoordT y

  The Y coordinate of the point.

3.13.1 Detailed Description

template<typename CoordT = double>struct frivol::Point< CoordT >

Two-dimensional point.

Template Parameters

CoordT	The coordinate type to use. Should be default constructible. Defaults to double, which is the coordinate type of DefaultPolicy.

3.13.2 Constructor & Destructor Documentation

3.13.2.1 template<typename CoordT = double> frivol::Point< CoordT >::Point (CoordT *x*, CoordT *y*) [inline]

Constructs point with given coordinates.

Parameters

x	The X coordinate.
y	The Y coordinate.

The documentation for this struct was generated from the following file:

- /home/topi/unison/Asiakirjat/frivol/frivol/point.hpp

3.14 frivol::Policy< CoordT, EventPriorityQueueT, BeachLineSearchTreeT > Struct Template Reference

`#include <policy.hpp>`

Public Types

- typedef CoordT **Coord**

3.14.1 Detailed Description

template<typename CoordT, template< typename PriorityT > class EventPriorityQueueT, template< typename ElementT > class BeachLineSearchTreeT>struct frivol::Policy< CoordT, EventPriorityQueueT, BeachLineSearchTreeT >

Policy class for the Fortune's algorithm, specifying data types and data structures to use.

Template Parameters

CoordT	The coordinate type to use. Should be ordered and default constructible to undefined value. Should have specialization of GeometryTraits.
EventQueueT	The priority queue type for events. Must conform to PriorityQueueConcept.
BeachLineT	The search tree to use for the "beach line" of arcs. Must conform to SearchTreeConcept.

The documentation for this struct was generated from the following file:

- /home/topi/unison/Asiakirjat/frivol/frivol/policy.hpp

3.15 frivol::containers::PriorityQueueConcept< X, PriorityT > Class Template Reference

`#include <priority_queue_concept.hpp>`

3.15.1 Detailed Description

template<typename X, typename PriorityT>class frivol::containers::PriorityQueueConcept< X, PriorityT >

Concept checking class for priority queues X with priority values of type PriorityT (or NIL). Priority queues are initialized with given size, and contain priority values for keys 0, 1, ..., size-1. Initially, all priority values are NIL. X must support the following operations:

- <construct>(Idx size) creates priority queue for keys 0, 1, ..., size-1.
- bool empty() const returns true if all keys have NIL priority.
- std::pair<Idx, PriorityT> pop() returns pair of a key with lowest non-NIL priority and its priority and sets the priority to NIL.
- void setPriority(Idx key, PriorityT priority) sets the priority value of 'key' to non-NIL value 'priority'.
- void setPriorityNIL(Idx key) sets the priority value of key 'key' to NIL.

X may assume that PriorityT is ordered with <-operator. X may have undefined behavior if supplied keys are out of range or if pop() is called when empty() returns true.

The documentation for this class was generated from the following file:

- /home/topi/unison/Asiakirjat/frivol/frivol/containers/priority_queue_concept.hpp

3.16 frivol::containers::SearchTreeConcept< X, ElementT > Class Template Reference

`#include <search_tree_concept.hpp>`

Public Types

- typedef X::Iterator **IteratorT**

3.16.1 Detailed Description

template<typename X, typename ElementT>class frivol::containers::SearchTreeConcept< X, ElementT >

Concept checking class for search trees X for elements of type ElementT. Search trees are sequence containers, the elements of which are iterated using iterator objects of type X::Iterator. The iterator must be a standard bidirectional iterator. X must support the following operations:

- <construct>() creates empty search tree.
- bool empty() const retuns true if the search tree is empty.
- Iterator begin() returns the iterator of the first element (or past-the-end if empty).
- Iterator end() returns the iterator past the last element.
- template<typename FuncT> Iterator search(FuncT func) searches the sequence using the supplied int(-Iterator)-function that for given iterator iter returns negative if the searched element is before iter, positive if it is after iter, and 0 if iter is the right element. If an element such that func returns 0 is found, it is returned, otherwise end() is returned.
- void erase(Iterator iter) removes element at iter. Other iterators should not be invalidated.
- Iterators insert(Iterator iter, const ElementT& elem) inserts elem before iter and returns the iterator of the new element. Does not invalidate any iterators.

X may assume that ElementT is copy constructible.

The documentation for this class was generated from the following file:

- /home/topi/unison/Asiakirjat/frivol/frivol/containers/search_tree_concept.hpp

3.17 frivol::containers::Stack< T > Class Template Reference

`#include <stack.hpp>`

Public Member Functions

- Stack ()

  Constructs empty stack.
- bool empty () const

  Returns true if the stack is empty.
- T & top ()
- void pop ()

  Removes the top element of the stack. Call only if empty() is false.
- void push (const T &element)

3.17.1 Detailed Description

template<typename T>class frivol::containers::Stack< T >

Stack of elements. The container constructs elements in advance so they should be default-constructible.

Template Parameters

T	The type of stored elements.

3.17.2 Member Function Documentation

3.17.2.1 template<typename T> void frivol::containers::Stack< T >::push (const T & *element*)

Pushes element to the top of the stack.

Parameters

element	The element to push.

3.17.2.2 template<typename T > T & frivol::containers::Stack< T >::top ( )

Returns reference to the top element of the stack. Call only if empty() is false.

The documentation for this class was generated from the following files:

- /home/topi/unison/Asiakirjat/frivol/frivol/containers/stack.hpp
- /home/topi/unison/Asiakirjat/frivol/frivol/containers/stack_impl.hpp

3.18 frivol::VoronoiDiagram< CoordT > Class Template Reference

`#include <voronoi_diagram.hpp>`

Classes

- struct **Edge**

Public Types

- typedef Point< CoordT > **PointT**

Public Member Functions

- VoronoiDiagram (Idx faces)
- Idx getFaceCount () const

  Returns the number of faces in the diagram.
- Idx getEdgeCount () const

  Returns the number of half-edges in the diagram.
- Idx getVertexCount () const

  Returns the number of Voronoi vertices in the diagram.
- Idx getFaceBoundaryEdge (Idx face) const
- Idx getTwinEdge (Idx edge) const
- Idx getIncidentFace (Idx edge) const
- Idx getStartVertex (Idx edge) const
- Idx getEndVertex (Idx edge) const
- Idx getNextEdge (Idx edge) const
- Idx getPreviousEdge (Idx edge) const
- const PointT & getVertexPosition (Idx vertex) const
- std::pair< Idx, Idx > addEdge (Idx face1, Idx face2)
- Idx addVertex (const PointT &pos, Idx edge1, Idx edge2, Idx edge3)
- void consecutiveEdges (Idx edge1, Idx edge2)

3.18.1 Detailed Description

template<typename CoordT>class frivol::VoronoiDiagram< CoordT >

Structure for storing a Voronoi diagram. The diagram consists of faces (one for each input site), edges between faces and Voronoi vertices (the endpoints of edges).

One vertex is always incident to three edges - vertices with four or more incident edges can be achieved by using duplicate voronoi vertices and edges of length 0. Each edge is actually a pair half-edges (twin edges) that represent the sides of the edge. The half-edges are thought to be directed so that they cycle around a face counterclockwise.

The faces, half-edges and Voronoi vertices are identified by numerical IDs 0...count-1. The ID of the faces should be the same as their corresponding input site indices.

3.18.2 Constructor & Destructor Documentation

3.18.2.1 template<typename CoordT > frivol::VoronoiDiagram< CoordT >::VoronoiDiagram (Idx *faces*)

Constructs Voronoi diagram.

Parameters

faces	Number of faces.

3.18.3 Member Function Documentation

3.18.3.1 template<typename CoordT > std::pair< Idx, Idx > frivol::VoronoiDiagram< CoordT >::addEdge (Idx *face1*, Idx *face2*)

Adds a new edge (two twin half-edges) to the Voronoi diagram.

Parameters

face1,face2	The IDs of the faces incident to the edge.

Returns

the IDs of the new half-edges, first one having face1 and the second one having face2 as incident face.

3.18.3.2 template<typename CoordT > Idx frivol::VoronoiDiagram< CoordT >::addVertex (const PointT & *pos,* Idx *edge1,* Idx *edge2,* Idx *edge3*)

Adds a new Voronoi vertex.

Parameters

pos	Position of the vertex.
edge1,edge2,edge3	The half-edges having the new vertex as end vertex, in counterclockwise order.

Returns

the ID of the new vertex.

3.18.3.3 template<typename CoordT > void frivol::VoronoiDiagram< CoordT >::consecutiveEdges (Idx *edge1,* Idx *edge2*)

Mark half-edges as being consecutive. Done automatically by addVertex for non-infinite ends of edges.

Parameters

edge1,edge2	The IDs of the half-edges such that edge2 should be next from edge1.

3.18.3.4 template<typename CoordT > Idx frivol::VoronoiDiagram< CoordT >::getEndVertex (Idx *edge*) const

Returns the ID of the Voronoi vertex in the end of given half-edge. If the diagram is incomplete and the vertex has not yet been found, nil_idx is returned.

Parameters

edge	ID of the half-edge.

3.18.3.5 template<typename CoordT > Idx frivol::VoronoiDiagram< CoordT >::getFaceBoundaryEdge (Idx *face*) const

Returns the ID of a half-edge that is on the boundary of given face. If the diagram is incomplete and no such edges have been found or there is only one face, nil_idx is returned.

Parameters

face	ID of the face that the half-edge should be incident to.

3.18.3.6 template<typename CoordT > Idx frivol::VoronoiDiagram< CoordT >::getIncidentFace (Idx *edge*) const

Returns the ID of the incident face of given half-edge.

Parameters

edge	ID of the half-edge.

3.18.3.7 template<typename CoordT > Idx frivol::VoronoiDiagram< CoordT >::getNextEdge (Idx *edge*) const

Returns the ID of the next half-edge around the incident face. If the diagram is incomplete and the next edge has not yet been found, nil_idx is returned.

Parameters

edge	ID of the half-edge.

3.18.3.8 template<typename CoordT > Idx frivol::VoronoiDiagram< CoordT >::getPreviousEdge (Idx *edge*) const

Returns the ID of the previous half-edge around the incident face. If the diagram is incomplete and the previous edge has not yet been found, nil_idx is returned.

Parameters

edge	ID of the half-edge.

3.18.3.9 template<typename CoordT > Idx frivol::VoronoiDiagram< CoordT >::getStartVertex (Idx *edge*) const

Returns the ID of the Voronoi vertex in the start of given half-edge. If the diagram is incomplete and the vertex has not yet been found, nil_idx is returned.

Parameters

edge	ID of the half-edge.

3.18.3.10 template<typename CoordT > Idx frivol::VoronoiDiagram< CoordT >::getTwinEdge (Idx *edge*) const

Returns the ID of the twin half-edge of given half-edge. Twin half-edges are the sides of an edge, having opposite directions.

Parameters

edge	ID of the half-edge.

3.18.3.11 template<typename CoordT > const Point< CoordT > & frivol::VoronoiDiagram< CoordT >::getVertexPosition (Idx *vertex*) const

Returns the position of a Voronoi vertex.

Parameters

vertex	ID of the Voronoi vertex.

The documentation for this class was generated from the following files:

- /home/topi/unison/Asiakirjat/frivol/frivol/voronoi_diagram.hpp
- /home/topi/unison/Asiakirjat/frivol/frivol/voronoi_diagram_impl.hpp